IUT 'A' Paul SABATIER
Dépt Informatique

ASR ⇒ Réseaux : Res2

TP 2 : Étude de Protocoles de niveau applicatif
Service de Transfert de fichiers FTP (RFC959)

---

1. Introduction

FTP est un service standard d'Internet pour le transfert de fichiers. Il est important de faire la différence entre le **transfert de fichiers**, qui est réalisé par FTP, et l'**accès aux fichiers** à travers un réseau qui est fourni par des applications telles que **NFS** (**N**etwork **F**ile **S**ystem de Sun). Le transfert de fichier consiste à recopier un fichier complet d'un système à un autre. Pour utiliser FTP nous devons posséder un compte sur le système distant pour pouvoir nous y connecter, ou nous devons l'utiliser avec un serveur qui autorise le FTP anonyme.

Comme Telnet, FTP a été conçu dès l'origine pour fonctionner entre des machines différentes, exécutant des systèmes d'exploitation différents, utilisant des structures de fichiers différentes et éventuellement des jeux de caractères différents. Alors que Telnet utilise un seul standard (l'ASCII NVT) auquel doivent se conformer les deux machines, FTP gère toutes les différences entre les systèmes en supportant un nombre limité de types de fichiers (ASCII, binaires...) et de structures de fichiers (à flux d'octets ou orientés enregistrements).

2. Le protocole FTP

FTP est un protocole orienté connexion. C'est à dire qu'il y a trois étapes :

- Établir une session entre le site local et le site distant
- Effectuer les traitements désirés
- Terminer la session

FTP diffère du modèle Client/Serveur standard car il utilise deux connexions TCP pour transférer un fichier :

- Une **connexion de contrôle** est utilisée pour acheminer les commandes (ou requêtes) du client vers le serveur et les réponses de protocole du serveur vers le client.
- Une **connexion de transfert de données** qui est créée à chaque fois qu'un fichier est transféré entre le client et le serveur (elle peut être créée à d'autres moments comme nous le verrons plus tard).

La figure suivante montre la configuration du client et du serveur et les deux connexions :

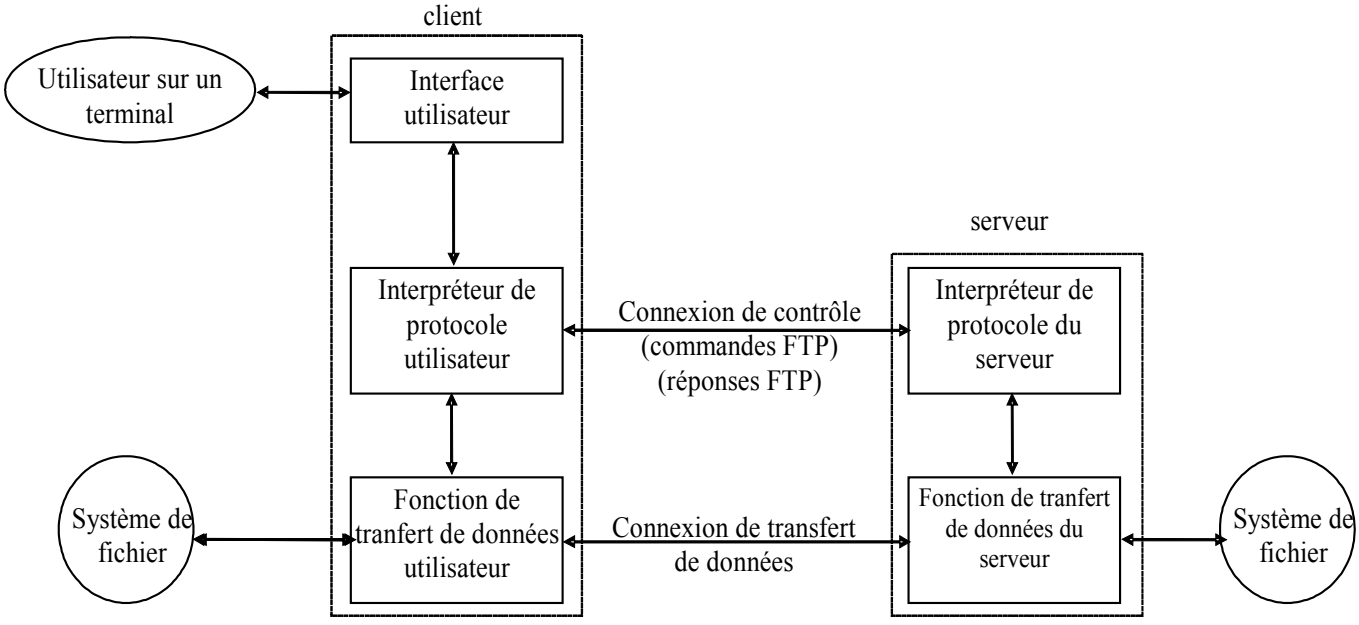


Figure 1 : Fonctions d'un transfert de fichiers

Cette figure montre que l'utilisateur n'a pas à s'occuper des commandes et réponses qui sont échangées le long de la connexion de contrôle. Ces détails sont gérés par les interpréteurs de protocole. L'élément appelé « interface utilisateur » gère le type d'interface utilisé pour gérer le transfert de fichiers : interface en fenêtre, en mode ligne ou à partir d'un programme (script shell par exemple).

Le schéma montre aussi que ce sont les interpréteurs de protocole qui invoquent les deux fonctions de transfert de données quand cela est nécessaire.

Pour la connexion de contrôle, le numéro de port utilisé par le serveur FTP est 21. C'est un port réservé (*well-known port*) qui est décrit dans le fichier `/etc/services`. Le client, quand à lui obtient un numéro de port **dynamiquement**.

Pour la connexion de transfert de données, le numéro de port utilisé par le serveur FTP est 20 (lui aussi un port réservé, sauf dans le mode passif ou le port est créé dynamiquement). Le client, quant à lui obtient, ici aussi, un numéro de port dynamiquement.

2.1. Représentation des données

De nombreux choix de représentation des données sont fournis par la spécification du protocole FTP. Les seuls qui sont encore utilisés aujourd'hui sont les types ASCII et binaires (BINARY ou IMAGE).

On utilise le mode ASCII lorsque les fichiers échangés ne contiennent que des caractères éditables. Dans ce cas, il y a prise en compte des différences entre les machines (traduction des séquences de retour à la ligne notamment). Les fichiers exécutables sont en mode binaire. En général c'est l'utilisateur qui doit choisir le mode de transfert (par défaut c'est en général le mode ASCII qui est utilisé). Dans le doute, il vaut toujours mieux utiliser le mode binaire.

Pour accélérer la récupération des données il vaut mieux compresser les fichiers à transmettre. On peut utiliser la commande `compress` (`uncompress` pour décompresser) — fichiers `.Z`—, `gzip` (`gunzip`) — fichiers `.gz`— sous Unix ou `pkzip` (`pkunzip`) — fichiers `.zip` — sous DOS/Windows. Certains serveurs ftp comme WU_FTPD gèrent la compression sur demande. C'est à dire que l'on peut demander qu'un fichier non compressé, soit d'abord compressé avant d'être transmis.

Pour transférer un répertoire entier (ou de nombreux fichiers) on peut utiliser un utilitaire d'archivage comme `tar` pour former un seul fichier. Certains utilitaires de compression peuvent aussi gérer cet archivage. C'est le cas du programme `winzip` qui est très répandu dans le monde PC.

Note : les fichiers compressés et/ou archivés doivent être transmis en mode binaire.

2.2. Commandes et réponses du protocole FTP

Les commandes et les réponses sont envoyées sur la connexion de contrôle en ASCII NVT (7 bits). Elles nécessitent à chaque fin de ligne une paire <CR, LF>.

2.2.1. Les commandes

Les commandes sont constituées de 3 ou 4 octets représentant des lettres en caractères ASCII. Certaines ont des arguments optionnels. Plus de 30 commandes sont définies. Le tableau suivant présente les commandes les plus utilisées :

Commande	*Description*
`ABOR`	avorte la commande FTP précédente et tout transfert de fichier en cours
`LIST liste de fichiers`	liste des fichiers ou des répertoires
`PASS mot de passe`	envoie le mot de passe au serveur
`PORT n1,n2,n3,n4,n5,n6`	adresse du client IP (n1.n2.n3.n4) et port (n5x256+n6)
`QUIT`	déconnexion
`RETR nom de fichier`	récupère un fichier du serveur
`STOR nom de fichier`	envoie un fichier au serveur
`SYST`	demande le type de système du serveur
`TYPE type`	spécifie le type de transfert : A pour ASCII, I pour image (binaire)
`USER nom d'utilisateur`	envoie le nom de l'utilisateur (son login) au serveur

Dans les exemples suivants, nous verrons qu'il y a souvent une correspondance de un à un entre les commandes tapées par l'utilisateur et les commandes envoyées au serveur. Cependant, certaines commandes utilisateur peuvent générer l'envoi de plusieurs commandes sur la connexion de contrôle.

2.2.2. Les réponses

Les réponses sont des nombres de trois chiffres en ASCII, avec un message optionnel suivant le nombre. En fait, le logiciel n'a besoin que du nombre pour déterminer comment traiter la réponse, et la chaîne optionnelle n'est là que pour faciliter la compréhension de l'utilisateur.

Chacun des trois chiffres du code réponse a une signification différente. Le tableau suivante présente la signification des premiers et seconds chiffres :

Réponse	*Description*
`1yz`	Réponse préliminaire positive. L'action est démarrée mais le client espère une autre réponse avant d'envoyer une autre commande.
`2yz`	Réponse complète positive. Une nouvelle commande peut être envoyée.
`3yz`	Réponse intermédiaire positive. La commande a été acceptée mais une autre commande doit être envoyée.
`4yz`	Réponse complète négative transitoire. L'action de la requête n'a pas eu lieu mais la condition d'erreur est temporaire de sorte que la commande doit être renvoyée plus tard.
`5yz`	Réponse complète négative permanente. La commande n'a pas été acceptée et ne devra pas être réessayée.
`x0z`	Syntaxe.
`x1z`	Information
`x2z`	Connexions. Réponses concernant les connexions de contrôle ou de données.
`x3z`	Authentification et accréditation. Réponses pour le login et les commandes de compte.
`x4z`	Non spécifié.
`x5z`	Etat du système de fichiers

Le troisième chiffre donne une signification supplémentaire à la réponse.
Voici quelques réponses typiques et les messages possibles :

- 125 Connexion de données déjà ouverte ; démarrage du transfert
- 200 Commande correcte.
- 214 Message d'aide (pour un utilisateur humain).
- 331 Nom d'utilisateurs correct, mot de passe requis.
- 425 Ne pas ouvrir la connexion de données.
- 452 Erreur d'écriture de fichier.
- 500 Erreur de syntaxe (commande non reconnue).
- 501 Erreur de syntaxe (arguments invalides).

2.2.3. Exemple de FTP

Exemple simple de FTP en mode -d (debug) qui permet de visualiser les requêtes qui sont envoyées par le client :

1- Connexion au serveur *ftp* de la machine `onyx.local` . Le serveur répond qu'il est prêt à traiter les commandes.

```
$ ftp -d onyx.local
```

```
Connected to onyx.local
220 onyx.local FTP server ................... ready
```

2- Le client nous demande un nom de login et un mot de passe (non retourné en écho).

```
Name (onyx.local:benzekri): benzekri
---> USER benzekri
```

```
331 Password required for benzekri.
Password:
---> PASS XXXX
230 User benzekri logged in.
```

3- Le client demande le type du système distant pour déterminer le mode de transfert par défaut (ici binaire).

```
---> SYST
215 UNIX Type: L8
Remote system type is UNIX.
Using binary mode to transfer files.
```

4- Demande de changement de répertoire qui est acceptée.

```
ftp> cd sock
```

```
---> CWD sock
250 CWD command successful.
```

5- Demande de la liste des fichiers du répertoire courant sur le site distant.

***Remarque 1* : la commande utilisateur ls se transforme en fait en deux commandes du protocole FTP.**
***Remarque 2* : le résultat de la commande LIST est envoyé en utilisant la connexion de transfert de données qui est spécialement créée pour la durée du transfert.**

```
ftp> ls
```

```
---> PORT 10,2,0,2,6,68
200 PORT command successful.
---> LIST
150 Opening ASCII mode data connection for /usr/bin/ls.
total 8
-rwxr-xr-x   1 benzekri   enseignants   1519 Jan 31  1995 ccux.c
-rwxr-xr-x   1 benzekri   enseignants   1964 Jan 31  1995 scux.c
226 Transfer complete.
```

6- Récupération d'un fichier distant qui met en œuvre trois commandes.

```
ftp> get ccux.c
```

```
---> PORT 10,2,0,2,6,71
200 PORT command successful.
---> RETR ccux.c
150 Opening BINARY mode data connection for ccux.c (1519 bytes).
226 Transfer complete.
1519 bytes received in 0.00 seconds (3071.22 Kbytes/s)
```

7- Envoi d'un fichier local au serveur.

```
ftp>  put scux.c
```

```
---> PORT 10,2,0,2,6,73
200 PORT command successful.
---> STOR scux.c
150 Opening BINARY mode data connection for scux.c.
226 Transfer complete.
336 bytes sent in 0.00 seconds (532.67 Kbytes/s)
```

8- Passage au mode passif.

```
ftp> Passive

ftp> ls

---> PASV
227 Entering passive mode (10,2,0,7,8,41)
---> LIST
150 Opening ASCII mode data connection for /usr/bin/ls.
total 8
-rwxr-xr-x   1 benzekri   enseignants   1519 Jan 31  1995 ccux.c
-rwxr-xr-x   1 benzekri   enseignants   1964 Jan 31  1995 scux.c
226 Transfer complete.
```

9- Arrêt de la connexion.

```
ftp> close

---> QUIT
221 Goodbye.
```

10- Sortie du programme ftp et retour au shell.

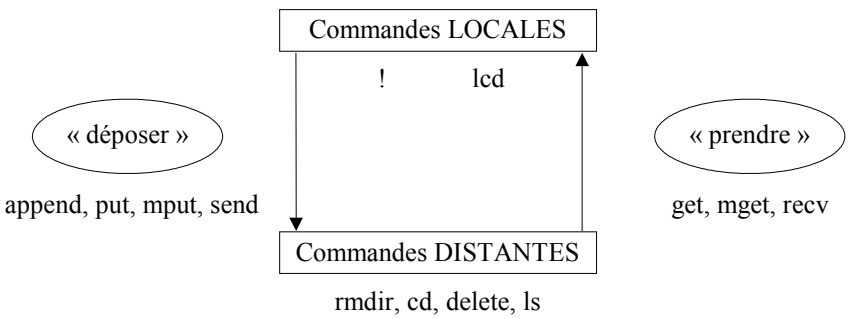
```
ftp> quit
$
```

Pour obtenir de l'aide sur les commandes de **ftp** on peut taper la commande **help** [nom de commande].

```
$ ftp
ftp> help
Commands may be abbreviated.  Commands are:

!               debug           mget            pwd             status
$               dir             mkdir           quit            struct
account         disconnect      mls             quote           system
append          form            mode            recv            sunique
ascii           get             modtime         reget           tenex
bell            glob            mput            rstatus         trace
binary          hash            newer           rhelp           type
bye             help            nmap            rename          user
case            idle            nlist           reset           umask
cd              image           ntrans          restart         verbose
cdup            lcd             open            rmdir           ?
chmod           ls              prompt          runique
close           macdef          proxy           send
cr              mdelete         sendport        site
delete          mdir            put             size
ftp> help append
append          append to a file
```

Quelques commandes FTP :



Changement de mode : commande **ascii** pour transférer en mode ASCII ; commande **binary** pour transférer en binaire.

Les problèmes courants que vous pouvez rencontrer dans une session FTP sont les suivants :

- « `Login incorrect` » : le login que vous avez tapé n'existe pas ou le mot de passe est incorrect.
- « `Unknown host` » : la machine distante n'existe pas.
- « `Host unreachable` » : la machine distante n'est pas accessible (problème de routage).
- « `Host not responding` » : la machine distante ne répond pas (problème sur la machine).
- « `Connection timed out` » : si vous ne tapez pas de commande pendant un long moment, le serveur FTP ferme la connexion.
- « `No such file or directory` » : le fichier ou le répertoire demandé n'existe pas.
- « `access denied` » : vous n'avez pas le droit de récupérer le fichier.

2.2.4. Exemple de FTP anonyme

Certaines machines proposent des serveurs FTP publics sur lesquels n'importe quel utilisateur peut se connecter pour récupérer et/ou déposer des fichiers. C'est ce que l'on appelle un **serveur FTP anonyme**.

```
$ ftp ftp.cict.fr

Connected to oceane.cict.fr.
220 oceane.cict.fr FTP server (Version wu-2.4.2-academ[BETA-14](1) Mon Sep 15
16:15:46 MET DST 1997) ready.
```

Note : au lieu de taper un login et un mot de passe on tape **anonymous** pour login et notre **adresse email** comme mot de passe.

```
Name (ftp.cict.fr:benzekri): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
```

Ensuite la session est normale.

```
ftp> cd pub/pc/win95/netutil

250 CWD command successful.

ftp> ls

200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 128028
drwxrwxr-x   2 ftpadm   cict        1024 Sep 18 07:09 .
drwxrwxr-x   7 ftpadm   cict         512 Dec 17  1996 ..
-rw-rw-r--   1 ftpadm   cict       57307 Jan  5  1996 descrpt2.exe
-rw-rw-r--   1 ftpadm   cict     2060851 May  3  1996 eudor154.exe
-rw-rw-r--   1 ftpadm   cict     4235500 Jun 27 14:26 eul301.exe
....
-rw-rw-r--   1 ftpadm   cict      201213 Jan 31  1996 ws_ftp32.zip
-rw-rw-r--   1 ftpadm   cict       57314 Feb  6  1996 wsping32.zip
-rw-rw-r--   1 ftpadm   cict     1829942 Sep 26  1995 xdemo32.zip
-rw-rw-r--   1 ftpadm   cict         629 May  3  1996 xwin32.ini
226 Transfer complete.

ftp> get ws_ftp32.zip

local: ws_ftp32.zip remote: ws_ftp32.zip
200 PORT command successful.
150 Opening BINARY mode data connection for ws_ftp32.zip (201213 bytes).
226 Transfer complete.
201213 bytes received in 0.67 seconds (291.72 Kbytes/s)

ftp> bye
221 Goodbye.
```

2.2.5. Utilisation de proxy FTP

Un **proxy FTP** est un logiciel qui agit comme intermédiaire (on dit aussi mandataire) entre un client FTP et un serveur FTP. Celui-ci se trouve à la frontière de 2 domaines, le domaine local et le reste du monde. Ainsi lorsque l'on veut accéder à un serveur FTP sur Internet, on peut être amené à utiliser un logiciel proxy FTP.
L'identification de l'utilisateur et du site serveur doit être communiquée au proxy sous la forme suivante :

```
nom_login@nom_serveur
```

Les commandes FTP peuvent alors être utilisées comme si l'on s'adressait directement au serveur FTP.
Il est à noter que les administrateurs de réseaux préfèrent utiliser des pare-feux orientés état (**stateful inspection firewalls**) plutôt que des logiciels proxy.

2.3. Exercices

Les réponses aux différentes questions devront être notées sur une feuille qui sera relevée à la fin du TP.

1) Lancer le client **ftp** avec l'option **debug** pour une connexion avec le serveur ftp de **onyx.local.**
   Noter les commandes/réponses échangées entre le client et le serveur ftp suite aux commandes de l'utilisateur.
2) Quand on se connecte sur un serveur FTP en utilisant la commande `ftp -d onyx.local`, les messages suivants s'affichent à l'écran :

   ```
   Connected to onyx.local
   220 onyx.local FTP server ............. ready
   Name (onyx.local:benzekri):
   ```

   Qui est à l'origine de ces messages ?

3) Ouverture des connexions de données : Dans le mode normal (actif), quelle entité ouvre la connexion de données ? Et dans le mode passif ? Établir le scénario correspondant à la session FTP ci-dessous :

   Connexion au serveur *ftp* de la machine `onyx.local` . Le serveur répond qu'il est prêt à traiter les commandes.

   ```
   $ ftp -d onyx.local
   ```

   ```
   Connected to onyx.local
   220 onyx.local FTP server ............. ready
   ```

   Le client nous demande un nom de login et un mot de passe (non retourné en écho).

   ```
   Name (onyx.local:benzekri): benzekri
   ---> USER benzekri
   331 Password required for benzekri.
   Password:
   ---> PASS XXXX
   230 User benzekri logged in.
   ```

   Passage au mode passif et liste des fichiers du répertoire distant.

   ```
   ftp> passive
   ftp> ls
   ```

   ```
   ---> PASV
   227 Entering passive mode (10,2,0,2,8,41)
   ---> LIST
   150 Opening ASCII mode data connection for /usr/bin/ls.
   total 8
   -rwxr-xr-x   1 benzekri   enseignants   1519 Jan 31  1995 ccux.c
   -rwxr-xr-x   1 benzekri   enseignants   1964 Jan 31  1995 scux.c
   226 Transfer complete.
   ```

   Arrêt de la connexion.

   ```
   ftp> close
   ```

```
---> QUIT
221 Goodbye.
```

Le scénario doit identifier les différents acteurs (l'utilisateur, le programme client et le programme serveur) et les messages échangés entre ces acteurs.

4) Depuis un client **telnet**, accéder au serveur FTP. Un utilisateur peut-il se connecter et ainsi établir la connexion de contrôle ?

   On souhaite lister le répertoire distant en mode actif. Que faut-il faire ? Que se passe-t-il ? Expliquer pourquoi ce n'est pas possible.

   Refaire la même chose en mode passif (scénario de la question 3). Pour cela tiliser deux programmes clients **telnet**.

   Rappel : Un client telnet peut se connecter à un serveur donné et à un port donné pour échanger des messages.

   La syntaxe d'appel est :

   `telnet` *`machine_serveur  port_serveur`*

5) Établir une connexion ftp anonyme sur le serveur ftp.cict.fr par l'intermédiaire du proxy FTP qui s'exécute sur **onyx.local** et qui écoute sur le port **2122**.
   Dans une autre fenêtre, ouvrir un connexion ssh sur onyx et noter les adresses IP et les numéros de port des connexions entre le client et le proxy et entre le proxy et le serveur.
   Représentez à l'aide d'un schéma les connexions que doit gérer un **proxy** lors d'une session FTP avec échange de fichiers.